NGSP – A Novel Parallel Signal Processing Architecture

D. Bruce Oakley[1], Richard Vallee[1], Gavin Hemphill[2], Robert Trider[3], F.Andrew Reid[1]

1. Applied Microelectronics, Inc., 1046 Barrington St., Halifax, Nova Scotia, B3H 2R1 Canada
2. Defence Research Establishment Atlantic, 9 Grove St., Dartmouth, Nova Scotia, B2Y 3Z7 Canada
3. Trider Scientific, 629 Windmill Rd., Unit 9, Dartmouth, Nova Scotia, B3B 1B7 Canada

Abstract

The Next Generation Signal Processor (NGSP) architecture was developed for real-time signal processing applications. A novel symmetric multiprocessor (SMP) architecture is presented which allows the overall system performance to scale linearly with the number of processing elements constituting the system. The architecture provides coherent caching of shared control memory on all processing elements, and a collision-free memory access scheme for common shared data memory.

Introduction

The NGSP architecture, illustrated in Figure 1, was developed for the Canadian Navy as a replacement for the AN/UYS-501 signal processor. The NGSP architecture maintained key features of the AN/UYS-501, including a contention-free shared working memory, high-speed cache and single cycle execution of transcendental functions. In addition, the new architecture offered MIMD capability.

The target application was the Experimental Towed Array Sonar System (ETASS) program. This program had processing requirements that could only be achieved using a multiprocessor. The need for deployment at sea called for an embedded signal processor. However, the developmental nature of the program also created a requirement for an easy to use programming environment. The NGSP was designed to meet these divergent needs.

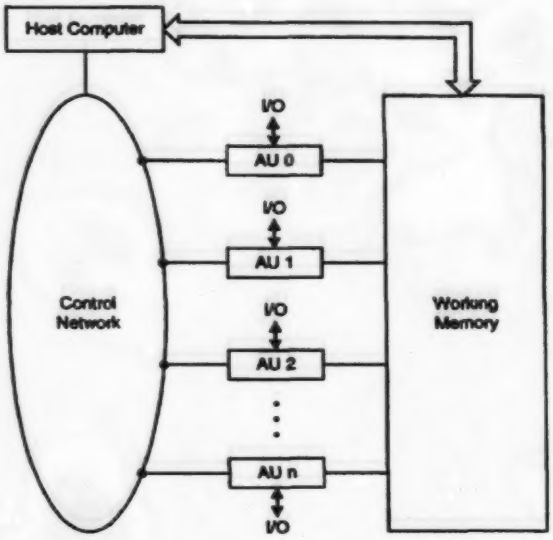

Figure 1: NGSP System Architecture

Traditionally, parallel signal processing architectures have been plagued with programming complexities and common memory sharing limitations, particularly as the number of parallel processing elements increases. Thus, the overall performance benefits of each processing element steadily decrease with the number of processing elements added, to the point where a net decrease in performance can be seen. In this paper, a novel SMP architecture is presented which allows the overall system performance to scale linearly with the number of processing elements constituting the system. The architecture provides

coherent caching of shared control memory on all processing elements, as well as a collision-free memory access scheme for common shared memory.

Background - Symmetric Multiprocessors

SMP systems are characterized mainly by the nature of their shared memory. Memory is shared uniformly among the processors, each of which has an identical view of the memory system. This uniformity implies that all processors have a consistent image of what is contained in memory, that is, the memory system is *coherent*.

This memory coherence eases the software burden of inter-processor communication, and makes it possible to run a single copy of an operating system on all processors concurrently. Furthermore, any task can be run on any processor. Scheduling and load distribution are controlled by the operating system. When effectively implemented, this model can greatly simplify development of application code, or porting such code from a single processor environment.

A shared memory system can be implemented by placing the actual memory in a central location, separated from all of the processors. This uniform memory access (UMA) model provides equal latency to all memory locations for all processors. Recently, the trend has been towards non-uniform memory access (NUMA) models, in which part of the shared memory is located at each processor.

Maintaining memory coherence is complicated by the use of local caches to hide memory access latencies. For coherence to be maintained, any changes to locally cached data must be reflected throughout the global memory system. Techniques for dealing with this abound, but mainly fall within three classes:

- **Snooping:** Hardware associated with each processor monitors, or snoops, all accesses to the memory system. This hardware can then update or invalidate its local copy when necessary. Snooping schemes are feasible when all processors access memory via a common bus. Many of today's microprocessors include snooping hardware in their bus interfaces.

- **Directory-Based:** The state of each cache line is maintained at each caching processor, as well as at the memory itself. This information is typically structured as a linked list, which removes the "broadcast" requirement associated with snooping. Only processors which actually hold a copy of a memory location are informed of updates. This allows the use of more generic interconnection methods in the memory system. Directory-based coherence has been used to implement cache coherent versions of the non-uniform memory access model (ccNUMA).

- **Software Methods:** Software methods exist for maintaining memory coherence, but these tend to detract from the ease of use of the simple, uniform memory model.

The ease of use afforded by the coherent memory model is one of the key advantages of the SMP approach. Unfortunately, it is generally achieved at the expense of scalability. Access to a globally shared memory can form a bottleneck when the number of processors becomes large.

A summary of the main characteristics of typical SMP systems:

- Ease of programming is paramount, supported by the shared memory model and general purpose processors.
- Meant for general purpose use, rather than targeted at specific applications.
- Extensive scalability is of secondary importance, and is difficult with global shared memory. Systems using NUMA tend to be more scalable, but software is burdened with optimization of memory allocation.

Background – Embedded Signal Processors

Embedded signal processors generally exploit application-specific knowledge to optimize performance. Extensive scalability has been achieved by matching memory structures to known, fixed data flows. Ease of programming has been of secondary importance, since application programs are generally only written once.

Traditionally, multiprocessors for DSP applications have often used distributed memory architectures, requiring message passing schemes for inter-processor communication. This reflects the emphasis on performance over ease of use. More recently, shared memory systems based on the NUMA model have been used in some systems. Hardware cache coherence, however, hasn't been broadly adopted.

Embedded signal processors are used in real time applications, which have strict requirements for completion of processing tasks. This implies that processing times must be predictable.

A summary of the main characteristics of typical embedded signal processors:

- Performance efficiency is emphasized. The goal is to optimize processing performance per unit of power, cost and size.
- Targeted at a specific, known application.
- Scalability is important, particularly for high performance applications such as sonar and radar.

The NGSP Architecture

The NGSP architecture was designed to overcome some of the common problems encountered in signal processing systems. The architecture was intended to be applicable to general signal processing problems, although particular attention was paid to sonar applications. Design goals included the following:

- **Scalability with Sustained Memory Bandwidth:** Memory access times in a multiprocessor system will increase when different processors contend for access at a single memory bank. This gets worse as processors are added to the system. By ensuring that different processors never try to access any bank at the same time, access times can be sustained as processors are added
- **Access Pattern Independence:** Memory systems often rely on interleaving to improve access times. This is effective when consecutive accesses are always targeted at different memory banks. However, it can be difficult to ensure this when access stride patterns change. This is a common problem when data is loaded in "row

order" and processed in "column order" (corner turning). Average access times to a multi-bank memory system can be reduced if bank distribution is made independent of the access pattern.

- **Fast Computation of Transcendental Functions:** Computation of trigonometric functions, for example, can be exceptionally slow. When the angles are known, such as those used in an FFT calculation, the values can be computed ahead of time and stored. However, some processing tasks, such as beam forming, involve angles that aren't known in advance.

- **Shared Memory System Programming Model:** Programming of system-level operations, such as task management, will be easier if a shared memory model is used for inter-processor communication.

NGSP is based on a system-level Harvard architecture, as shown in the block diagram in Figure 1. Data is stored in the Working Memory, while code and control information pass over the Control Network. More specific details are presented in the following sections.

Working Memory

This is a high-bandwidth, UMA memory system which is scalable with sustained access bandwidth. The Working Memory is implemented using a rotating crossbar network and a proprietary deterministic address sequencing scheme. The address sequencing ensures sequential accesses to separate physical banks regardless of the logical memory stride. As a result, the rotating crossbar network provides collision-free access to all processors regardless of their number.

Control Network

The Control Network is a ccNUMA shared memory system, implemented using SCI, the Scalable Coherent Interface (IEEE Std 1596-1992).

Arithmetic Units

Actual processing is performed by Arithmetic Units (AUs), illustrated in Figure 2. The main components of an AU are:

- **Vector Engine:** The Vector Engine is a highly pipelined, custom signal processor. It is optimized for vector operations, and has an instruction set which uses vector operands. All operations, including transcendental functions, have single-cycle throughput.

- **Multi-ported Cache Memory:** This memory is five-way interleaved, and uses an address hashing technique to distribute access patterns across physical banks. The Vector Engine is capable of simultaneously performing three operand reads and two operand writes in a single cycle.

- **Data Transfer Unit:** This is essentially a programmable, multi-ported, queued DMA server. The Data Transfer Unit can move data between the Working Memory, Cache Memory, Control Network and I/O Interface.

- **I/O Interface:** This is mezzanine based, to support different application requirements.

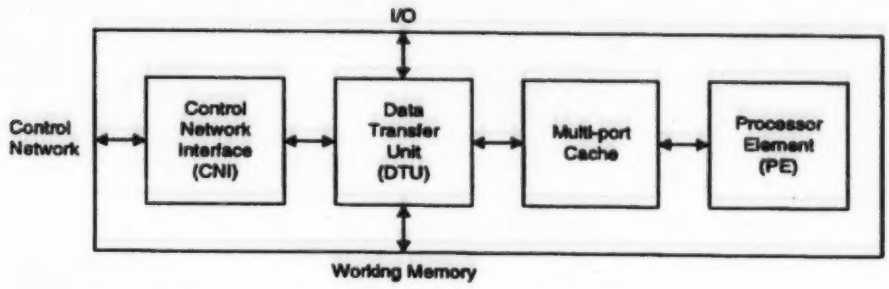

Figure 2: High Level Description of the Arithmetic Unit

Host Computer

The Host is intended to be an off-the-shelf single board computer, providing system-level supervision and a user interface. It must interface to the Control Network via SCI, and have a VME interface to the Working Memory for setup operations.

Results

An experimental prototype NGSP system has been built, and deployed with towed-array sonar on Canadian destroyers. Using early to mid 1990's technology, the following performance was realized:

- 800 MFLOPS peak, 400 MFLOPS sustained, per AU
- 160 MBytes/s sustained access to Working Memory for each AU

The coherent Control Network forms the basis of a coherent, real-time kernel. Based on this, an Integrated Programming Environment (IPE) has been developed. Applications developed in this environment have shown linear performance scaling up to the largest system built to date (8 AUs).

References

Lenoski, D.E. & Weber, W.-D., *Scalable Shared-Memory Multiprocessing*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1995

Jaenicke, R., "A NUMA Architecture for DSP using the PowerPC", *International Conference on Signal Processing Applications and Technology*, pg. 1416-1420, Sept. 14-17, 1997

Jaenicke, R., "Selecting a Multiprocessor DSP Architecture", *DSP World Spring Design Conference*, pg. 447-462, Apr. 21-23, 1998

Stenstrom, P., "A Survey of Cache Coherence Schemes for Multiprocessors", *IEEE Computer*, Vol. 23, No. 6, pg. 12-24, June 1990

James, D.V., "The Scalable Coherent Interface: Scaling to High-Performance Systems", Compcon 94, Vol. 25, pg. 64-71, IEEE Computer Society Press, Los Alamitos, CA, 1994

Vallee, R., et al., "NGSP – Experimental Development Model (XDM)", International Conference on Signal Processing Applications and Technology, Sept. 13-16, 1998

Pilkington, C., et al., "Overview of the NGSP Integrated Programming Environment", International Conference on Signal Processing Applications and Technology, Sept. 13-16, 1998